Knowl Inf Syst
DOI 10.1007/s10115-012-0559-9

REGULAR PAPER

Supervised term weighting centroid-based classifiers for text categorization

Tam T. Nguyen · Kuiyu Chang · Siu Cheung Hui

Received: 9 March 2012 / Revised: 28 April 2012 / Accepted: 22 August 2012

Abstract In this paper, we study the theoretical properties of the class feature centroid (CFC) classifier by considering the rate of change of each prototype vector with respect to individual dimensions (terms). We show that CFC is inherently biased toward the larger (dominant majority) classes, which invariably leads to poor performance on class-imbalanced data. CFC also aggressively prune terms that appear across all classes, discarding some non-exclusive but useful terms. To overcome these CFC limitations while retaining its intrinsic and worthy design goals, we propose an improved centroid-based classifier that uses precise term-class distribution properties instead of presence or absence of terms in classes. Specifically, terms are weighted based on the Kullback–Leibler (KL) divergence measure between pairs of class-conditional term probabilities; we call this the CFC–KL centroid classifier. We then generalize CFC–KL to handle multi-class data by replacing the KL measure with the multi-class Jensen–Shannon (JS) divergence, called CFC–JS. Our proposed supervised term weighting schemes have been evaluated on 5 datasets; KL and JS weighted classifiers consistently outperformed baseline CFC and unweighted support vector machines (SVM). We also devise a word cloud visualization approach to highlight the important class-specific words picked out by our KL and JS term weighting schemes, which were otherwise obscured by unsupervised term weighting. The experimental and visualization results show that KL and JS term weighting not only notably improve centroid-based classifiers, but also benefit SVM classifiers as well.

Keywords Centroid classification · Support vector machines · Kullback–Leibler divergence · Jensen–Shannon divergence

T. T. Nguyen (✉) · K. Chang · S. C. Hui
School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore
e-mail: nguy0080@e.ntu.edu.sg

K. Chang
e-mail: askychang@ntu.edu.sg

S. C. Hui
e-mail: asschui@ntu.edu.sg

Published online: 09 September 2012

1 Introduction

In this paper, we study the theoretical properties of the simple and efficient centroid-based classifier, which works as follows. During the training or learning phase, a centroid is computed for each class. During prediction, a test sample is classified to the class of the nearest centroid. Guan et al. [8] recently proposed a class feature centroid (CFC) classifier that uses intra-class and inter-class term presence/absence to boost the weight of highly discriminative terms during training. Each term is weighted by its document frequency (the intra-class information) and a class discriminative factor that is inversely proportional to the number of classes containing it (the inter-class information). In addition, CFC goes against conventional wisdom by not normalizing the centroid for prediction, invariably boosting the weights of exclusive words originating from larger classes. CFC was evaluated on the Reuters-21578 and 20-Newsgroup datasets and shown to significantly outperform the simple arithmetical average centroid (AAC) and cumuli geometric centroid (CGC) approaches as well as the well-known SVM baseline classifier.

While the CFC has been shown to work very well on multi-class datasets in general, it suffers from two major disadvantages: (1) it performs very poorly for binary class data due to the aggressiveness in which it penalizes non-discriminatory terms, that is, those that appear in both classes, regardless of the actual counts and distribution and (2) it is inherently biased toward the majority class, that is, it performs poorly on highly imbalanced datasets.

To illustrate, consider a two-class dataset (positive and negative classes) where a term appears in both positive and negative documents. In CFC, this term will be assigned zero weight by virtue of its inter-class commonality. Essentially, what CFC is doing is a very aggressive form of feature selection; a term is useful if and only if it appears exclusively in one class, or in as few numbers of classes as possible in the multi-class case. One consequence of this aggressive term weighting philosophy is that outlier terms like people names can become overly important, as will be shown later in the example in Table 1.

The second limitation of CFC is that it does not work well on imbalanced datasets, which we shall prove theoretically. For prediction, CFC uses the de-normalized cosine similarity function to find the nearest centroid to an unlabeled document. This essentially means that highly discriminative terms, that is, those with exclusive class document frequency, will dominate the similarity comparison. A natural effect of this bias is that larger classes will have larger weights for every class-exclusive term.

Borrowing the moment generating function from multi-set theory and applying it to the class document frequency (number of documents containing a term within the class), we show that the CFC grows exponentially with respect to the class document frequency, in contrast to the linear growth of CGC and AGC. As such, we propose a way to regulate the exponential CFC growth with a discriminatory multiplier based on the mixture proportion of term probabilities within the two classes (for binary datasets). For multi-class datasets, we subsequently proposed a generalized Jensen–Shannon (JS) divergence metric to approximate this regulation across multiple classes.

Table 1 Term counts and CFCs for C_1 and C_2

Feature	C_1	C_2	$\mathbf{c}_1$	$\mathbf{c}_2$
t_1	100	1	0	0
t_2	1	100	0	0

2 Related work

Centroid-based classifiers are fast, efficient, and easy to deploy. In [9], Han and Karypis analyzed and presented a linear-time centroid-based text classification algorithm, which outperformed a number of classical classifiers including Naïve Bayes [15,17], K-Nearest-Neighbor [5], and C4.5 Decision Tree [27]. Building upon Han's work, Guan et al. [8] subsequently proposed the class feature centroid (CFC) classifier, which boosts the importance of highly discriminative terms. The CFC has been evaluated on multi-class textual datasets and shown to even beat the state-of-the-art support vector machine (SVM) classifier. In their comparison, CFC and SVM were both configured to use the same feature space, namely $tf \times idf$ (term-frequency-inverse-document-frequency) [20]. The unsupervised term weighting $tf \times idf$ is widely used in information retrieval, originally based on the ranking function BM25. A newer variant is BM25t [7] (the "t" stands for tags) for XML documents, which weigh words based on the relevance of their associated XML tags.

Although the CFC algorithm works very well in text categorization, it can be further improved by adjusting the importance of each term/dimension in a supervised setting. Specifically, the centroid in the classical CFC classifier has two major components: the term frequency (local component computed for each document) and the inverse document frequency (global component computed across the entire corpus). For supervised term weighting, the idf global component is replaced by some class-specific weight.

Debole and Sebastiani [6,28] previously proposed a supervised term weighting scheme $tf \times \chi^2$, which uses the chi-square (χ^2) term statistic to boost highly discriminative terms. In their approach, the idf global component is simply replaced by the χ^2 statistic of the term. Generally, the χ^2 statistic is very small if a term is independent of the class and vice versa.

In [14], Lan et al. proposed another supervised term weighting method called $tf \times rf$ ("rf" stands for relevance frequency), which is a two-class extension to $tf \times idf$. Here, the idf component is replaced by rf. Intuitively, $tf \times rf$ tries to give higher weightage to terms that appear more frequently in the positive class. However, there are some peculiarities with $tf \times rf$. First, $tf \times rf$ is asymmetric because it favors terms that appear more in the "positive" class only. This means that in practical two-class problems, the choice of positive class will have an impact on the term weighting and ultimately the classification performance. Second, $tf \times rf$ does not penalize terms that appear equally often in both classes. In the above two special cases, terms are simply assigned their tf weights.[1]

In $tf \times \delta idf$ (delta idf) [21], Martineau and Finin replaced the idf with the class inverse document frequency difference, namely delta inverse document frequency. The importance of a term is proportional to the disparity of its distributions across the positive and negative classes. In other words, the term weight equals zero if it is evenly distributed among the two classes. On the other hand, it gets larger with increasingly skewed term-class distributions. Our approach shares the same goal of exploiting the class-distribution differences, except that we characterize the distributional difference using divergences, which is mathematically more elegant.

The aforementioned supervised term weighting techniques have been applied to various text categorization tasks including sentiment analysis [4,10,11], email spam filtering [13,19], news classification, spyware detection [16], and job title classification [2].

Visualization of text data help users explore and analyze a large collection of documents. Wei et al. [30] developed a text visualization system based on topic analysis and interactive

[1] In $tf \times rf$, uniformly distributed terms, that is, those that appear equally in both classes, are assigned a constant weight of $1.58tf$.

visualization techniques. To visualize word clouds, we propose another approach that is based on the term weights and class distributions, focusing specifically on sentiment analysis. In fact, sentiment analysis was the original motivation for our term weighting approach. It is a special case of text categorization where the textual data are classified into two (positive or negative) or three categories (positive, negative, or neutral).

3 Centroid-based classification

Given a set of N training document vectors $\{\mathbf{x}_n : n = 1, \ldots, N; \mathbf{x}_n \in \mathbb{R}^D\}$ and the corresponding class labels $\{y_n : n = 1, \ldots, N; y_n \in \{1, \ldots, M\}\}$, the goal (during the training phase) is to determine M *centroids*, $\{\mathbf{c}_m : m = 1, \ldots, M; \mathbf{c}_m \in \mathbb{R}^D\}$ corresponding to each of the M categories. The centroid $\mathbf{c}_m$ is also known as the *prototype vector* for class C_m. Let $C_m = \{\mathbf{x}_n \mid y_n = m\}$ denote the set of all vectors of class m.

For prediction, an unlabeled document $\mathbf{x}$ is assigned to the class of the closest centroid, based on a similarity measure $\text{sim} : \{(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_i \in \mathbb{R}^D, \mathbf{x}_j \in \mathbb{R}^D\} \rightarrow \{s \in \mathbb{R} \mid 0 \leq s \leq 1\}$

$$\hat{y}(\mathbf{x}) = \underset{i=1,m}{\text{argmax}} \ \text{sim}(\mathbf{x}, \mathbf{c}_i) \tag{1}$$

The most common similarity measure used for text classification is the cosine similarity [20].

Many methods have been proposed to compute the class prototype vectors, of which the two earliest ones are arithmetical average centroid (AAC) and cumuli geometric centroid (CGC) [9]. In these approaches, the prototype vectors are the arithmetical average or cumulative sum of all document vectors within a class. For instance, the AAC and CGC approaches calculate the centroids, respectively, as

$$\mathbf{c}_m^{\text{AAC}} = \frac{1}{\mid C_m \mid} \sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n \tag{2}$$

and

$$\mathbf{c}_m^{\text{CGC}} = \sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n \tag{3}$$

where $m = 1, \ldots, M$.

For the CFC-based classification method [8], the d-th entry of the m-th prototype vector $\mathbf{c}_m^{\text{CFC}} = \left[c_{m,1}^{\text{CFC}}, \ldots, c_{m,D}^{\text{CFC}}\right]$ is computed as

$$c_{m,d}^{\text{CFC}} = b^{\frac{N^m(x_d)}{|C_m|}} \log\left(\frac{M}{\zeta(x_d)}\right) \tag{4}$$

where $N^m(x_d)$ is the document frequency of term x_d in class C_m and $\zeta(x_d)$ is the number of classes containing term x_d.

Intuitively, inter-class information can boost the importance of highly discriminative terms. However, assigning the term weight to zero when a term occurs in all classes, as espoused by CFC in the second part of (4), is quite drastic. A term may appear at different frequencies in different classes! To illustrate this problem, consider the following example.

Suppose we have a two-class dataset with class labels C_1 and C_2 and a corpus vocabulary of two terms t_1 and t_2. Each class contains two documents, that is, $C_1 = \{(20, 1), (80, 0)\}$ and $C_2 = \{(0, 10), (1, 90)\}$. Here, each document is represented as a term frequency row vector, for example, the first document in class 1 contains 20 occurrences of term 1 (t_1) and one

Fig. 1 Class word cloud of the movie review dataset. **a** Positive, **b** negative

occurrence of term 2 (t_2). Table 1 shows the distribution of terms and centroids as determined by CFC. Clearly, term t_1 is quite unique to class C_1, while term t_2 is unique to class C_2, but the CFCs for this extreme example is useless because they are all zero! CFC effectively issues the "death penalty" to any term that appears across (although in disproportionate frequencies) all classes, which is clearly an overkill.

3.1 Word cloud model

Figure 1 shows the word clouds[2] of positive and negative classes in a movie review dataset [25] (dataset D1 in Table 3). Common nouns like "movie", "film", and "story" clearly dominate both classes. In addition, polarity words such as "like", "good", and "funny" appear with similar frequencies (indicated by font size) in both classes! On the other hand, only a few words like "best" and "bad" appear more frequently in the positive and negative classes, respectively. This means that the use of raw unweighted polarity words alone may not be optimal for classifying documents into positive and negative classes. On the other hand, the classifier should intuitively benefit from a term weighting approach that boost highly discriminative words.

In this section, we present a word cloud model for text categorization, borrowing from multi-set theory. In this model, each class is represented by a multi-set of words, also known as a word cloud. We then determine the class of a document by calculating its "distance" to each class, and classifying it to the nearest class. Without loss of generality, we assume that each word in a document can be represented by an integer. We formalize the problem as follows:

Definition 3.1 (*Word Set*) A *Word Set* A of words $t_1, t_2, \ldots, t_n$ appearing in a corpus of documents is defined as

$$A = \{t_1, t_2, \ldots, t_n\}$$

where $t_i \in \mathbb{N}^+$. Note that $A \subset \mathbb{N}^+$.

Definition 3.2 (*Word Cloud*) A *Word Cloud* D of a set of documents is defined as

$$D = (A, \mu)$$

where $\mu : A \mapsto \mathbb{N}^+$ is a vector member function, and each $\mu(t_i)$ indicates the number of times $t_i \in A$ appears in D.

[2] Generated by Wordle (http://www.wordle.net).

Definition 3.3 (*Moment Generating Function*) A moment generating function $m : \mathbb{R}^A \mapsto \mathbb{R}^A$ of a word cloud $D = (A, \mu)$ is defined as

$$m_D(\mathbf{t}) = \mathbb{E}\left[\mathrm{e}^{\mathbf{t}^{\mathrm{T}}\mu(A)}\right]$$

where E[·] is the expectation (average) operator. Note that the moment generating function is unique for each corresponding word cloud. Taking advantage of its uniqueness property, we are able to use it to determine the class of documents by comparing the "distance" between document word cloud and class word cloud.

Theorem 3.1 (Union) *If $D_1, D_2, \ldots, D_n$ are word clouds with the corresponding moment generating functions $m_{D_i} = \mathbb{E}[e^{\mathbf{t}^{\mathrm{T}}\mu(A_i)}]$, then the moment generating function of the union $D = \bigcup_i D_i$ is defined as*

$$m_D(\mathbf{t}) = \prod_i m_{D_i}(\mathbf{t})$$

Proof Suppose that $D = (A, \mu(A))$, we have $A = \bigcup_i A_i$ and $\mu(D) = \sum_i \mu(A_i)$. Therefore, we have the moment generating function of D as follows:

$$m_D(\mathbf{t}) = \mathbb{E}\left[\mathrm{e}^{\mathbf{t}^{\mathrm{T}}\mu(A)}\right] = \mathbb{E}\left[\mathrm{e}^{\sum_i \mathbf{t}^{\mathrm{T}}\mu(A_i)}\right] = \prod_i \mathbb{E}\left[\mathrm{e}^{\mathbf{t}^{\mathrm{T}}\mu(A_i)}\right]$$

which concludes the proof. □

Corollary 3.1 (Mean Word Cloud) *If $D_1, D_2, \ldots, D_n$ are word clouds with the corresponding moment generating functions $m_{D_i} = \mathbb{E}[e^{\mathbf{t}^{\mathrm{T}}\mu(A_i)}]$, then we have their mean word cloud defined as*

$$\hat{D} = \left(\bigcup_i A_i, \frac{1}{n}\sum_i \mu(A_i)\right)$$

and its moment generating function defined as

$$m_{\hat{D}} = \mathbb{E}[e^{\frac{1}{n}\sum_i \mu(A_i)}]$$

Definition 3.4 (Word Cloud Similarity) Let D_1 and D_2 be two word clouds, whose similarity is defined as the cosine similarity between their moment generating functions as follows:

$$\mathrm{sim}(D_1, D_2) = \frac{\langle m_{D_1}(\mathbf{t}), m_{D_2}(\mathbf{t})\rangle}{\| m_{D_1}(\mathbf{t}) \| \| m_{D_2}(\mathbf{t}) \|}$$

where $\langle\cdot\rangle$ is the dot product. It is trivial to show that this similarity measure is a metric.

Word cloud classification can thus be summarized into the following three steps.

- *Document Representation*. Convert documents into word clouds and determine their moment generating functions.
- *Centroid Computation*. Determine the moment generating function of each class combining the moment generating functions of each member document.
- *Class Prediction*. Calculate the similarity between a document word cloud and all class word clouds based on their moment generating functions.

3.2 Centroid-based classifier with Kullback–Leibler divergence

In this section, we propose a centroid-based classifier called CFC–KL using the word cloud model.

3.2.1 Document representation

Similar to the bag-of-words vector space model [20], we first preprocess documents by converting them into word clouds. Suppose A is a set of all terms in the corpus, each word cloud can be represented by a member function $\mu(A)$. Clearly, $\mu(A)$ is a term frequency vector in this space. Here is where supervised term weighting can be applied.

Formally, each document is represented by a D-dimensional real vector where each dimension x is determined as follows:

$$w(x) = w_{local}(x) \cdot w_{global}(x).$$

The popular $tf \times idf$ weighting scheme [20] for term x is defined as

$$w_{tf \times idf}(x) = tf(x) \cdot \log \frac{N}{n(x)} \tag{5}$$

where $tf(x)$ is the number of times term x appears in the document, N is the total number of documents, and $n(x)$ is the number of documents containing term x. Here, $tf(x)$ is the local weight for term x and $w_{idf}(x) = \log \frac{N}{n(x)}$ is its global weight.

To improve the classification accuracy, Lan et al. proposed the $tf \times rf$ term weighting [14], which modifies the global component as follows:

$$w_{tf \times rf}(x) = tf(x) \cdot \log_2 \left(2 + \frac{n^+(x)}{\max\left(1, n^-(x)\right)} \right)$$

where $n^+(x)$ and $n^-(x)$ are the number of positive and negative class-labeled documents containing term x, respectively. Intuitively, $tf \times rf$ tries to give higher weightage to terms that appear more frequently in the positive class via the global component w_{rf}. This means that in practical two-class problems, the choice on which class to be positive will have an impact on the term weighting and ultimately the classification accuracy. Moreover, $tf \times rf$ does not penalize terms that appear equally often in both classes, missing out on a useful piece of information.

To overcome the aforementioned problems, we propose a more robust term weighting method based on the Kullback–Leibler (KL) divergence [1] between the probabilities of terms across each class.

Let the a priori class probabilities be defined as

$$P(+) = \frac{N^+}{N} \quad \text{and} \quad P(-) = \frac{N^-}{N}$$

where N^+/N^- is the number of positive/negative documents and N is the total number of documents. We have the joint probability of term x belonging to the positive, negative, and both classes as follows:

$$P(x,+) = \frac{n^+(x)}{N^+}$$
$$P(x,-) = \frac{n^-(x)}{N^-}$$
$$P(x) = \frac{n^+(x) + n^-(x)}{N}$$

where $n(x) = n^+(x) + n^-(x)$. From the above, we can derive the class-conditional probabilities of term x as follows:

$$P(+ \mid x) = \frac{n^+(x)}{N^+}\frac{N}{n(x)}$$
$$P(- \mid x) = \frac{n^-(x)}{N^-}\frac{N}{n(x)}$$

Finally, the Kullback–Leibler term weighting is determined as the difference between the Kullback–Leibler divergence from $P(+ \mid x)$ and $P(- \mid x)$ to $P(x)$ as follows:

$$w_{\mathrm{KL}}(x) \simeq \begin{cases} \bar{N}(x) \mid \log\left(\frac{n^+(x)}{n^-(x)}\frac{N^-}{N^+}\right) \mid & \text{if } n^+(x) \neq 0 \text{ and } n^-(x) \neq 0 \\ 1.0 & \text{otherwise} \end{cases} \tag{6}$$

where $\bar{N}(x)$ is the normalized document frequency. Note that we may replace $\bar{N}(x)$ with $\log(\bar{N}(x))$ to reduce the influence of document frequency on the global term weight, just as typically done for $tf \times idf$. And the new term weighting approach is an extension of the KL term weighting proposed in [23].

3.2.2 Learning the centroids

In this section, we propose a new approach to determine the mean word cloud of classes based on Corollary 3.1. Recall that we have a set of documents $\{\mathbf{x}_n : \mathbf{x}_n \in \mathbb{R}^D; n = 1, \ldots, N\}$. Suppose that $\mathbf{x}_n$ is also a term frequency vector. We have the moment generating function of the mean word cloud as follows:

$$m_{C_m} = \mathbb{E}\left[\mathrm{e}^{\frac{1}{|C_m|}\sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n}\right]$$

Similar to $tf \times$ KL term weighting, we use the KL term weight to boost the classification accuracy by determining the centroid of class C_m as follows:

$$c^m_{\mathrm{KL}} = \mathrm{e}^{\frac{1}{|C_m|}\sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n} \times \bar{N}(\mathbf{x}_n) \sum_{k=1}^{M-1}\sum_{l=k+1}^{M} \mid \log\left(\frac{n^k(\mathbf{x}_n)}{n^l(\mathbf{x}_n)}\frac{N^l}{N^k}\right) \mid \tag{7}$$

where M is the number of classes.

In binary classification problems, the above formula can be simplified as follows:

$$\mathbf{c}^+_{\mathrm{KL}} = \mathrm{e}^{\frac{1}{N^+}\sum_{\mathbf{x}_n \in C^+} \mathbf{x}_n} \times \bar{N}(\mathbf{x}_n) \mid \log\left(\frac{n^+(\mathbf{x}_n)N^-}{n^-(\mathbf{x}_n)N^+}\right) \mid$$

and

$$\mathbf{c}_{\mathrm{KL}}^{-} = \mathrm{e}^{\dfrac{1}{N^-}\displaystyle\sum_{\mathbf{x}_n \in C^-} \mathbf{x}_n} \times \bar{N}(\mathbf{x}_n) \mid \log\left(\frac{n^+(\mathbf{x}_n)N^-}{n^-(\mathbf{x}_n)N^+}\right) \mid$$

3.2.3 Prediction

During prediction, an unlabeled document is assigned the $tf \times$ KL weight as given in Eq. (6). We compute the similarity between the unlabeled vector against all centroids, using the similarity function defined in Definition 3.4, it is assigned the class with the highest similarity as described in Eq. (1).

3.3 Growth of class document frequency

We now examine the rate of change of each term in centroid $\mathbf{c}$ with respect to a change in the class term frequency. We consider each term separately, assuming independence of the terms.

Proposition 3.1 *Let t denote the class-specific document frequency for a term in class C with centroid $\mathbf{c}$. Then, each term $\{c_d \mid d = 1, \ldots, D\}$ in the AAC or CGC has a linear growth rate with respect to its class-specific document frequency t, that is, $dc/dt = -k/t$ where k is a constant that differs for each term.*

Proof Suppose each document in class C is weighted by $tf \times idf$, substituting Eq. (5) into the AAC or CGC function for a particular term x, we have

$$\mathbf{c} = \frac{1}{\mid C \mid} \sum_{\mathbf{x}_n \in C} f(x_n) \log \frac{N}{t} = \frac{1}{\mid C \mid} \log \frac{N}{t} \sum_{\mathbf{x}_n \in C} f(x_n) = \frac{L}{\mid C \mid} \log \frac{N}{t} \tag{8}$$

where the class document frequency $N(x)$ is replaced by t and the local weights $f(x_n)$ are collected into the constant $L = \sum_{\mathbf{x}_n \in C} f(x_n)$. Taking the derivative of Eq. (8) with respect to t, we obtain our result:

$$\frac{\mathrm{d}c}{\mathrm{d}t} = -\frac{L}{\mid C \mid t} = -\frac{k}{t} \tag{9}$$

where k is a constant for term x. The proof for CGC is trivially similar and therefore omitted. It can be easily shown that CGC has the same inverse growth rate with constant $k = L$. □

Proposition 3.2 *Each term $\{c_d \mid d = 1, \ldots, D\}$ in the CFC has an exponential growth rate with respect to its class-specific document frequency t, that is, $dc/dt = k_1 b^{k_2 t}$ where b is a base exponent, k_1 and k_2 are constants for each term.*

Proof First we rewrite Eq. (4) in terms of the class document frequency t as follows:

$$c = b^{\frac{t}{|C|}} \log\left(\frac{M}{\zeta(x_d)}\right) \tag{10}$$

Taking the derivative with respect to t yields our result:

$$\frac{dc}{dt} = \frac{1}{\mid C \mid} \log b \log\left(\frac{M}{\zeta(x_d)}\right) b^{\frac{t}{|C|}} = k_1 b^{k_2 t} \tag{11}$$

where $k_1 = \log b \log\left[M/\zeta(x_d)\right]/|C|$ and $k_2 = 1/|C|$. □

Table 2 Centroid growth versus class document frequency

	dc/dt	k_1	k_2
AAC	$-k_1/t$	$\frac{1}{\lvert C\rvert}\sum_{\mathbf{x}_n \in C} f(x_n)$	–
CGC	$-k_1/t$	$\sum_{\mathbf{x}_n \in C} f(x_n)$	–
CFC	$k_1 b^{k_2 t}$	$\frac{1}{\lvert C\rvert}\log b \log\left[M/\zeta(x_d)\right]$	$\frac{1}{\lvert C\rvert}$
CFC–KL	$k_1 \mathrm{e}^{k_2 t}$	$\bar{N}(x_d)\sum_i \sum_j \mid \log\left(\frac{N^i(x_d)+1}{N^j(x_d)+1}\frac{N^j}{N^i}\right) \mid$	$\frac{1}{\lvert C\rvert}$

What Eq. (11) tells us is that for every new document containing term x added to the centroid, the class document frequency t for term x will increase by 1, while the CFC value for term x will increase exponentially by $k_1 b^{k_2}$. On the contrary, the AAC or CGC will only increase by k. In the long run, the CFC will be dominated completely by terms that are exclusive to the class based on their exponential growth. This also means that centroids of smaller classes will grow exponentially slower (fewer class documents), thereby allowing the large classes to dominate the prediction.

Compared to CFC, our proposed CFC–KL approach takes the form of an e-folding time scale function. It has the same property as CFC but is simpler because the model parameter b need not be specified.

3.4 Growth rate summary

Table 2 summarizes the growth rate of a centroid term with respect to the class document frequency. We see that for AAC and CGC, the growth rate is actually negative, which means that as more documents are added to the class, the centroid becomes more and more stable, adjusting itself only slightly. This is expected since the average operation tends to have a smoothing effect as the number of samples grow larger.

3.5 Centroid-based classifier with Jensen–Shannon divergence

The KL divergence measures the "distance" between two probability distributions, and has been used quite extensively in practice apart from our use of it to compute term weights. However, it has a few limitations described as follows. First, it is not a true metric because it not symmetric. Second, it does not follow the triangle inequality. Third, although the KL divergence is a non-negative function, it is not bounded. For example, in the extreme cases when $n^k(\mathbf{x}_n) = 0$ or $n^l(\mathbf{x}_n) = 0$, the KL supervised term weight can approach infinity. Although smoothing techniques can be applied, the smoothed term weights may become intractable. These limitations led us to consider the more general Jensen–Shannon divergence for multiple classes. Let p_1 and p_2 be two probability distributions, then the Jensen–Shannon divergence is defined as follows:

$$\mathrm{JS}(p_1, p_2) = -(\pi_1 p_1 + \pi_2 p_2)\log(\pi_1 p_1 + \pi_2 p_2) + \pi_1 p_1 \log(p_1) + \pi_2 p_2 \log(p_2)$$

where $\pi_1 \geq 0$, $\pi_2 \geq 0$ are mixing parameters that sum to unity, $\pi_1 + \pi_2 = 1$.

For M classes, the Jensen–Shannon divergence is defined more generally as follows:

$$\mathrm{JS}(p_1, \ldots, p_M) = -\left(\sum_{i=1}^{M} \pi_i p_i\right)\log\left(\sum_{i=1}^{M} \pi_i p_i\right) + \sum_{i=1}^{M} \pi_i p_i \log(p_i)$$

where $\sum_{i=1}^{M} \pi_i = 1$ and $\pi_i \geq 0$ for all i.

Remarks In the extreme cases of $p_i = 0$ for all i, $p_i \log(p_i) = 0$ always hold because $\lim_{p_i \to 0} p_i \log(p_i) = 0$. Thus, in contrast to KL divergence, JS divergence works well in the extreme case and is symmetric. Moreover, according to Lin [18], JS divergence is bounded in $[0, 1]$. Compared to KL divergence, JS divergence uses a set of weights $\{\pi_i\}$ for the probability distributions $\{p_i\}$. By taking these weights into account, we are able to propose another term weighting approach that works well on multi-class-imbalanced datasets.

In this section, a new term weighting approach based on Jensen–Shannon divergence is formulated as follows.

3.5.1 Document representation

For the sake of simplicity, the JS term weighting approach is formulated for binary problems first. Similar to the KL term weighting approach $tf \times \mathrm{KL}$, the supervised term weight KL is replaced by JS, which is defined as follows. Let $p^+ = n^+(x)/N^+$ and $p^- = n^-(x)/N^-$ be the probability of term x belonging to the positive and negative classes, respectively. Then, the JS weight is

$$\begin{aligned} \mathrm{JS}(p^+, p^-) = &- \left(\pi^+ p^+ + \pi^- p^-\right) \log \left(\pi^+ p^+ + \pi^- p^-\right) \\ &+ \pi^+ p^+ \log(p^+) + \pi^- p^- \log(p^-) \end{aligned}$$

where $\pi^+ + \pi^- = 1$, $\pi^+ \geq 0$, and $\pi^- \geq 0$. We name the new term weighting approach $tf \times \mathrm{JS}$.

Remarks If the dataset is balanced, that is, uniformly distributed, then π^+ and π^- will be assigned to equal weights of 0.5. Otherwise, they should be inversely proportion to their class proportions, that is, larger classes should have smaller π and vice versa. Specifically, we have $\pi^+ = N^-/N = (N - N^+)/N$ and $\pi^- = N^+/N = (N - N^-)/N$.

3.5.2 Learning the centroids

Similarly, to determine the centroids, the KL term weight is replaced by the JS term weight as follows:

$$c_{\mathrm{JS}}^m = \mathrm{e}^{\frac{1}{|C_m|} \sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n} \times \mathrm{JS}(p_1, \ldots, p_M)$$

where M is the number of classes.

To calculate the JS term weights, the weights of the prior class probability distributions must be known in advance. Let N_i be the number of documents in class i, the weight π_i is determined based on the *softmax* function [22] as follows:

$$\pi_i = \frac{\mathrm{e}^{\frac{-N_i}{N}}}{\sum_{j=1}^{n} \mathrm{e}^{\frac{-N_j}{N}}}$$

where N is the total number of documents and n is the number of classes. It can be seen that π_i satisfies the conditions $\pi_i > 0$ and $\sum_i \pi_i = 1$.

Discussion Clearly, the JS divergence is bounded in $[0, 1]$ [18]. Therefore, the JS term weights are always non-negative and less than 1.0. Unfortunately, this also means that the term frequency weight, which is unbounded, may overshadow the JS term weight. To prevent this from happening, we simply normalize each centroid to unit length as follows.

Let $\bar{\mathbf{x}}_m = \frac{1}{|C_m|} \sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n$ be the mean vector for class m, the normalized centroid has the following form:

$$c_{\text{JS}}^m = \frac{\bar{\mathbf{x}}_m}{\|\bar{\mathbf{x}}_m\|} \times \text{JS}(p_1, \ldots, p_M)$$

For binary datasets, we have

$$\mathbf{c}_{\text{JS}}^+ = \frac{\bar{\mathbf{x}}^+}{\|\bar{\mathbf{x}}^+\|} \times \text{JS}(p^+, p^-)$$

and

$$\mathbf{c}_{\text{JS}}^- = \frac{\bar{\mathbf{x}}^-}{\|\bar{\mathbf{x}}^-\|} \times \text{JS}(p^+, p^-)$$

3.5.3 Prediction

In the prediction phase, we use the cosine similarity function to calculate the distance between a document vector and each centroid. The cosine similarity measure levels the playing field for documents of all lengths. We also normalize the centroid vector to unit length, as opposed to CFC, that is, $\frac{1}{N^+} \sum_{\mathbf{x}_n \in C^+} \mathbf{x}_n$ and $\frac{1}{N^-} \sum_{\mathbf{x}_n \in C^-} \mathbf{x}_n$ are replaced by $\frac{\sum_{\mathbf{x}_n \in C^+} \mathbf{x}_n}{\|\sum_{\mathbf{x}_n \in C^+} \mathbf{x}_n\|}$ and $\frac{\sum_{\mathbf{x}_n \in C^-} \mathbf{x}_n}{\|\sum_{\mathbf{x}_n \in C^-} \mathbf{x}_n\|}$, respectively. We call this new approach CFC–JS, keeping the CFC prefix for simplicity despite its deviation from the original CFC.

4 Experiments

We compare the performance of our proposed CFC–KL and CFC–JS with the baseline CFC for centroid classifiers. In addition, to see how KL and JS term weightings perform on popular classifiers, we also evaluate them on support vector machine (SVM) [12][3] classifiers weighted by $tf \times idf$ (called SVM–IDF), $tf \times rf$ (called SVM–RF), $tf \times$ KL (called SVM–KL), and $tf \times$ JS (called SVM–JS). All experiments were conducted using 5-fold cross-validation unless otherwise stated.

4.1 Dataset

We evaluate our proposed approaches on three binary datasets (movie review, sentiment polarity, and multi-domain sentiment) and two multi-class datasets (Reuters-21578 and 20-newsgroups), with details shown in Table 3. They are briefly discussed as follows:

– D1: Movie Review. The binary movie review dataset (version 2.0) from Pang and Lee [25] contains 1,000 positive and 1,000 negative average length movie reviews.

[3] We use the LibSVM (http://www.csie.ntu.edu.tw/cjlin/libsvm/) library with linear kernels and default parameters.

Table 3 Datasets

Dataset	Terms	N	N^+
D1: Movie review	36,911	2,000	1,000
D2: Sentiment polarity	15,912	10,662	5,331
D3: Multi-domain sentiment	473,456	8,000	4,000
D4: 20-Newsgroup*	26,214	–	–
D5: Reuters-21578*	18,933	–	–

* Multi-class

Fig. 2 IDF term weight visualization of the movie review dataset

- D2: Sentiment Polarity. The binary Sentiment Polarity dataset [26] contains 5,331 positive and 5,331 negative snippets of short sentences.
- D3: Multi-domain Sentiment. The multi-domain sentiment dataset (version 2.0) [3] contains 8,000 samples with 473,456 unigram and bigram features. This large dataset will showcase how well our weighting method performs on high dimensional and large datasets.
- D4: 20-Newsgroup. The 20-Newsgroup dataset contains 26,214 average length documents spanning 20 categories. The number of documents in each class is balanced. We use this dataset to evaluate our proposed approach on multi-class problem.
- D5: Reuters-21578. The Reuters-21578 dataset contains 8,293 documents belonging to 65 categories. Each document has about 8,293 terms. The number of documents in each category is not balanced. While the first category has 3,713 documents, some of the others contain just a handful documents.

Documents were processed without stemming or stop-word removal.

4.2 Term weighting visualization

Figures 2, 3, 4, and 5 show the word clouds of the four global term weighting methods idf, rf, KL, and JS, respectively. Here, the term colors correspond to the classes. Negative, positive, and neutral (appearing in both classes) terms are shown in red, blue, and green, respectively. Term sizes are proportional to their term weights. Horizontal term positions indicate the relative frequency in the class, for example, a term on the far right appears very frequently in the positive class.

From the word cloud plots, it is quite obvious to see that idf removes all stop words (e.g., the, is, an) by assigning their weights to zeros (and therefore invisible in the word cloud), as shown in Fig. 2. Unfortunately, it also eliminates exclusively negative and positive words like “brilliantly”, “unimaginative”, which cannot be found anywhere in the idf word

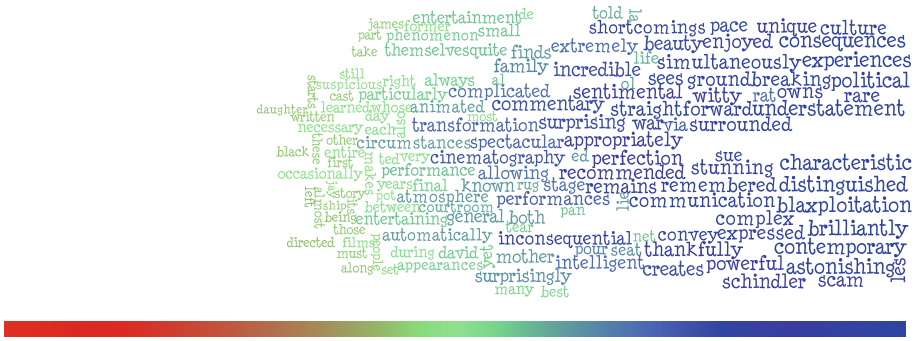

Fig. 3 RF term weight visualization of the movie review dataset

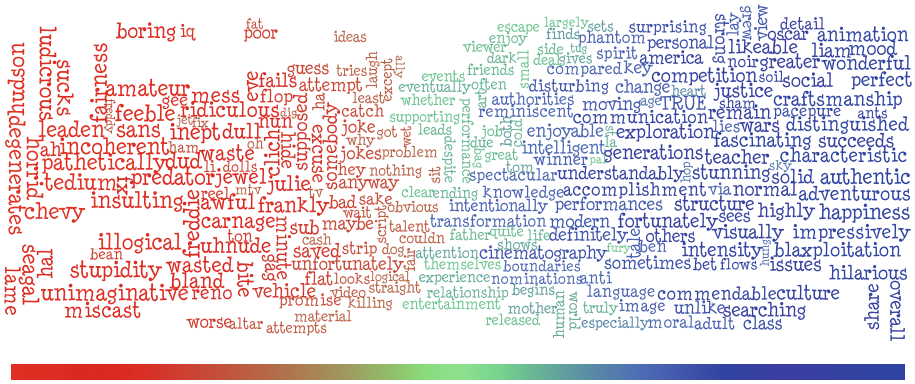

Fig. 4 KL term weight visualization of the movie review dataset

Fig. 5 JS term weight visualization of the movie review dataset

cloud. The vast majority of words of significant size/weight are in fact non-polarity words like “anthropologist”, “demonstration”,“ballroom”.

From the rf word cloud in Fig. 3, rf assigns larger weights to positive terms but tends to assign small constant weights to negative terms (invisible in the rf word cloud because they have relatively small weights).

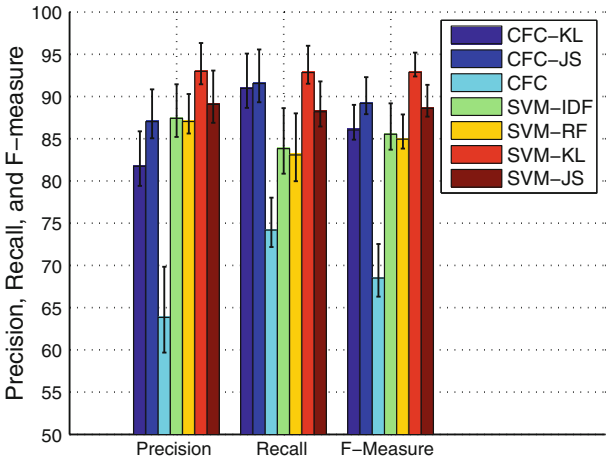

Fig. 6 Movie review dataset (D1) results

Figure 4 shows the KL term weight word cloud. It assigns higher weights to both exclusively negative and positive terms. For neutral terms in the middle, it tends to assign smaller weights. Take the highly negative word "horrid" as an example, which appears on the left-most part of the KL word cloud in a large font. Similarly, extremely positive words like "impressively" and "perfect" are shown on the far right. It can be seen visually that KL term weighting effectively boosts the importance of polarized terms in the movie review dataset.

The JS word cloud is shown in Fig. 5. The JS cloud is similar to the KL cloud in that it manages to capture the highly polarized words while assigning smaller weights for neutral terms. However, the difference in weight/size between polarized words and neutral words is not that significant because unlike the unbounded KL weights, JS weights are bounded to be less than or equal to 1.

4.3 Binary classification

We evaluate the proposed KL and JS weighting schemes on binary text classification datasets by plotting their 5-fold cross-validated F-measure, precision, and recall values along with error-bars (one standard deviation).

The results for the movie review dataset (D1) are shown in Fig. 6. We see that SVM–KL has the overall best F-measure at 93 %. This is followed by CFC–JS and SVM–JS vying for second place at around 89 % each. Although CFC–JS was only second best, it still managed to beat the baseline CFC by a whopping 20 % in terms of F-measure.

When it comes to SVM versus CFC, all flavors of SVM (SVM–KL, SVM–JS, SVM–IDF, SVM–RF) easily beat the baseline CFC. In fact, we could not replicate the superiority of CFC over SVM as claimed in [8] after several attempts. All in all CFC–JS managed to be better or similar to SVM–JS in terms of F-measure, winning on recall but losing out a little on precision.

If the absolute best F1 performance is demanded, then SVM–KL is the overall best choice. However, if the superior efficiency and incremental updating properties of CFC is desired, then CFC–JS is not a bad second choice. Thus, we see that JS weighting is suited for centroid classifiers, while KL weighting is more suitable for SVM classifiers. With JS or KL weighting, even CFC-based classifiers can meet or exceed the performance of a vanilla SVM-IDF.

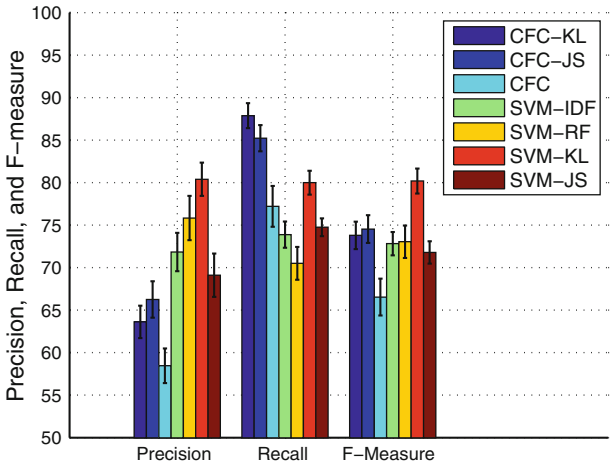

Fig. 7 Sentiment polarity dataset (D2) results

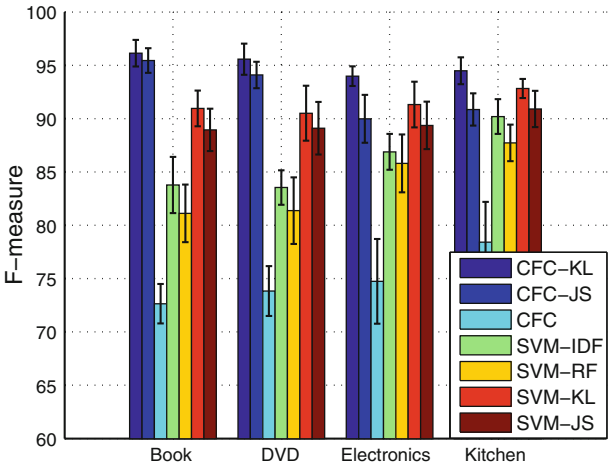

Fig. 8 F-measure for multi-domain sentiment dataset (D3)

This is significant because CFC classifiers are two to three orders of magnitude faster to train compared to SVM classifiers [24], and they can be updated incrementally. This may be very important considerations for the practical deployment of sentiment analysis algorithms in operational scenarios.

The results for the Sentiment Polarity dataset (D2) are shown in Fig. 7. Here, we see a similar trend where SVM–KL remains the leader with 80 % F1, with CFC–JS and CFC–KL in second place at 75 % F1, while the trio of SVM–IDF, SVM–RF, and SVM–JS nearly tied for third place. Interestingly, CFC–{KL, JS} both had very high precision of 87 and 85 %, respectively, compared to the second runner-up SVM–KL at 80 %. However, all three CFC methods scraped the bottom of the barrel in terms of precision. If consistency is desired, SVM–KL is the best performing classifier for D2, if high recall is important (e.g., in search engine applications where all relevant results are listed for the user to pick and choose), then CFC–JS and CFC–KL offer 5–7.5 % better recall, respectively.

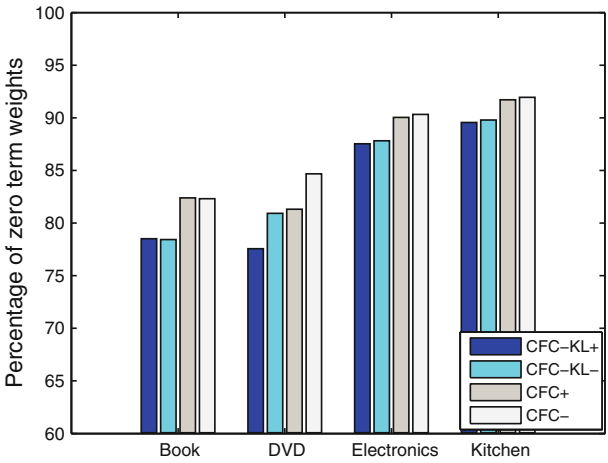

Fig. 9 Proportion of zero-weighted terms (%) in each (+/−) centroid for multi-domain sentiment dataset (D3)

For the Multi-domain Sentiment dataset (D3), the F-measure, recall, and precision results over 4 different domains (book, DVD, electronics, and kitchen) are shown in Fig. 8. From the F-measure results in Fig. 8, we see that CFC–KL has the best performance over all domains. The second best performer is splitted 50/50 between CFC–JS and SVM–KL, with SVM–JS consistently at fourth place. CFC performed the worst across all domains. Note that CFC–KL consistently achieved closed to 95 % F-measure across all domains. Both SVM–JS and SVM–KL are pretty consistent over all domains, achieving around 90 and 91 % F-measure, respectively. On the contrary, SVM–RF and SVM–IDF vary a lot across domains. We can conclude from this observation that JS and KL weights not only improve performance, but also help maintain a stable prediction performance across domains. To put it generally, JS and KL term weights consistently work well across different domains.

4.3.1 A view into centroids

As mentioned, CFC is very aggressive in feature selection; it will assign zero weight to terms appearing in both (positive and negative) classes, while exponentially growing the class-exclusive terms. To illustrate, we calculate the percentage of zero term weights in each CFC and CFC–KL centroids. The results are shown in Fig. 9.

We note that CFC–KL has approximately 3 % fewer zero-weighted terms compared to CFC on average. The difference between the CFC–KL and CFC in terms of zero term weights is the smallest for the kitchen domain, which happens to be the most separable case, for which both CFC and CFC–KL prune non-discriminatory terms aggressively (90–91 % zero-weighted terms). The Electronics domain is similar to the Kitchen domain, having close to 90 % zero-weighted terms.

Compared to CFC–KL, CFC has 5 % more zero-weighted terms for the book and DVD domains. This explains why CFC performed the worst and second-worst in the book and DVD domains, respectively; CFC simply discarded too many terms. In fact, CFC–KL maintained 5 % more terms than CFC by weighting them smartly, which translates handsomely to 20 % better F1 performance (as shown in Fig. 8).

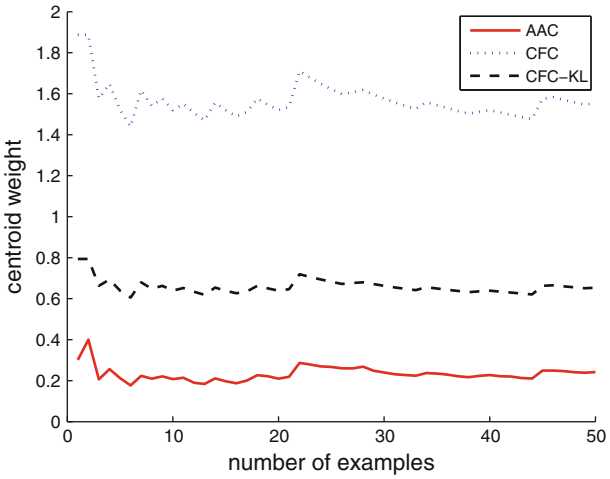

Fig. 10 Evolution of centroid term "like"

4.3.2 Growth rate of centroids

To illustrate the growth of prototype vectors, we plot the AAC, CFC, and CFC–KL weights of a class-specific centroid term "like" in Fig. 10, and for a generic term "movie" in Fig. 11. From the two figures, we see that the CFC–KL weight is generally two times larger than AAC, while CFC weights are four times larger than AAC. CFC weights also oscillate wildly compared to the other two. CFC–KL is more stable, while AAC achieved stability very early on. Whenever a document containing the centroid word appears, all three weights will spike, and then gradually decreases as new documents not containing the word are added. The next time it adds a new document containing the word, its centroid weight will spike again.

The interesting thing to note is that all three weights exhibit the same trends, differing only in their raw magnitudes. This means that in theory, we could scale up the weight of either one to obtain the other. The question to ask then is this, which scale gives the best performance? CFC is too aggressive, while AAC is too conservative. A delicate balance needs to be struck to find the right scale. CFC–KL clearly gives the best results as far as classification performance and stability is concerned.

4.4 One-off classification of multi-class-imbalanced data

Although the imbalanced dataset problem has been addressed using boosting techniques [29], term weighting techniques are simple yet effective solutions that can work equally well. To study the impact of imbalanced data on the overall performance of classifiers, we evaluate our proposed approach on imbalanced datasets, which are very common in practice. The imbalanced datasets are generated from the 20-newsgroup dataset where we keep one class as positive and the remaining classes as one big negative class. The details of the datasets are listed in Table 4, from which we see that each 20-newsgroup sub-dataset has a highly skewed distribution of around 4 % positive and 96 % negative data.

We trained CFC, CFC–KL, and CFC–JS classifiers on the 10 sub-dataset using 10-fold cross-validation. The experimental results are shown in Figs. 12 and 13 for the 20-newsgroup and Reuters-21578 datasets, respectively. Out of the 20 sub-datasets, CFC–JS leads 14, with

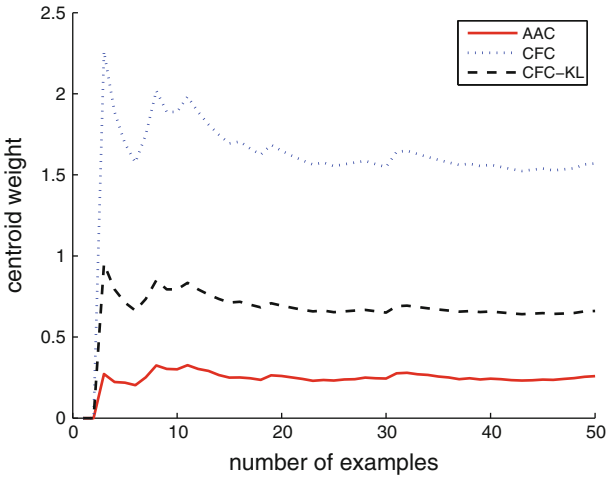

Fig. 11 Evolution of centroid term "movie"

Table 4 Imbalanced 20-Newsgroup (D4) and Reuters-21578 Datasets (D5)

Dataset	Terms	N^+	N^-
D4-1	26,214	799	18,047
D4-2	–	973	17,873
D4-3	–	985	17,861
...	...	...	...
D4-8	–	990	17,856
D4-9	–	996	17,850
D4-10	–	994	17,852
D5-1	473,456	3,713	4,580
D5-2	–	2,055	6,238
D5-3	–	321	7,972
...	...	...	...
D5-8	–	114	8,179
D5-9	–	110	8,183
D5-10	–	90	8,203

CFC–KL taking the crown for the remaining 6. Interestingly, whenever CFC–KL was the winner, CFC–JS was a very close runner-up but the same cannot be said when CFC–JS was in the lead. For example, in sub-datasets 5, 6, 7, and 10 of D5 in Fig. 13, CFC–JS beat CFC–KL by large margins varying from 15 to 47 %. This just goes on to show that CFC–JS is a more stable term weighting scheme to use for highly imbalanced datasets.

4.5 JS versus KL on imbalanced datasets

We study the pros and cons of JS term weighting vis-a-vis KL term weighting for SVM classification. Since we can adjust the class weights $\{\pi_i\}$ in the JS term weighting approach, it can work well on imbalanced datasets. To do that, we generate artificially imbalanced

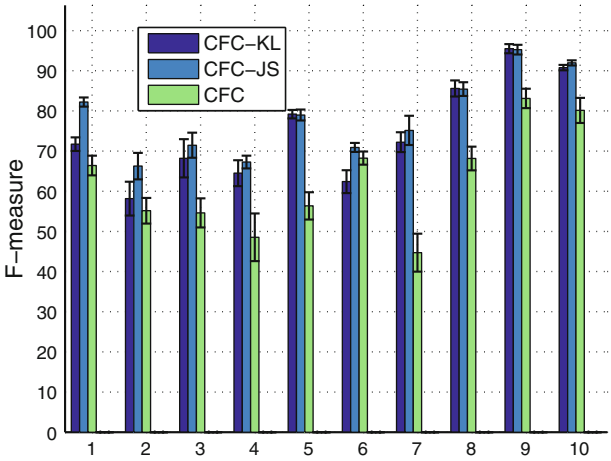

Fig. 12 F-measure for imbalanced 20-Newsgroup dataset (D4)

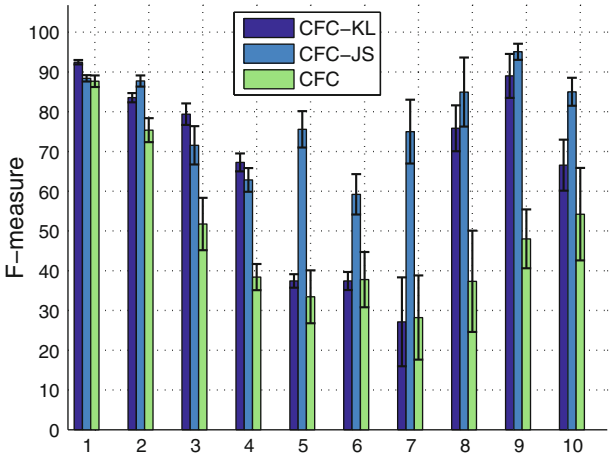

Fig. 13 F-measure for imbalanced Reuters-21578 dataset (D5)

datasets based on the originally balanced Multi-domain Sentiment dataset (D3), where the ratio of positive to negative training documents is gradually varied from 1:9 (100 positive 900 negative) to 9:1 (900 positive 100 negative), with 5:5 being a perfectly balanced training dataset. Figures 14, 15, 16 and 17 plot the F-measure ('+': positive, '−': negative) for the 4 domains in the Multi-domain Sentiment dataset (D3) versus varying ratios of positive to negative class training sizes.

From the positive class performances shown in Figs. 14a, 15a, 16a, and 17a, we can draw the following observations:

1. JS generally did better than or comparable to KL at the two extreme ends/ratios.
2. JS has a generally more consistent performance over all mixing proportions.
3. KL performed slightly better at the middle ground ratios of 4:6, 5:5, and 6:4. This is because KL weights are more aggressive than JS weights, which must lie in the unit interval. Thus, as the proportion approaches balance, KL weights lead to better results.

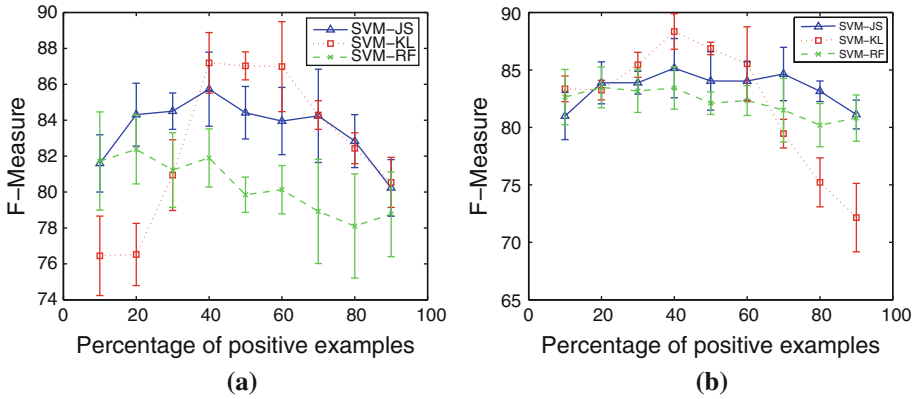

Fig. 14 F-measure for positive and negative classes on the book domain. **a** F-measure+, **b** F-measure−

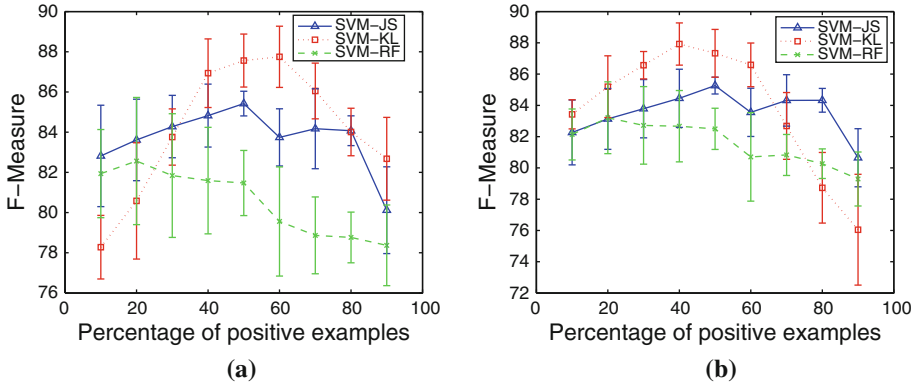

Fig. 15 F-measure for positive and negative classes on the DVD domain. **a** F-measure+, **b** F-measure−

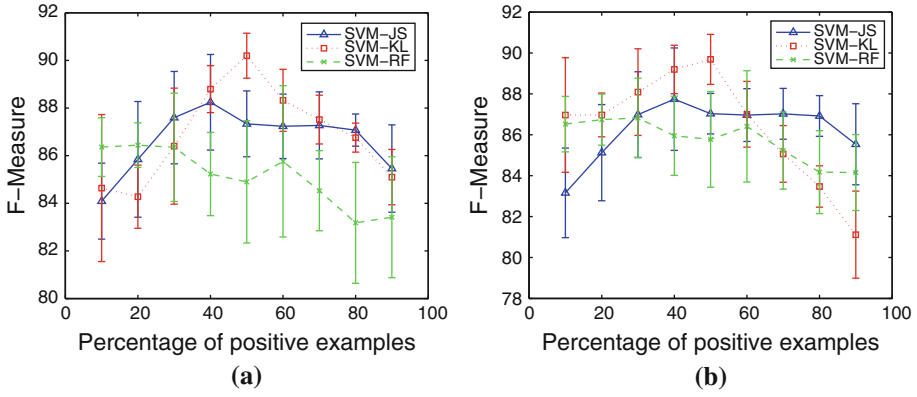

Fig. 16 F-measure for positive and negative classes on the electronics domain. **a** F-measure+, **b** F-measure−

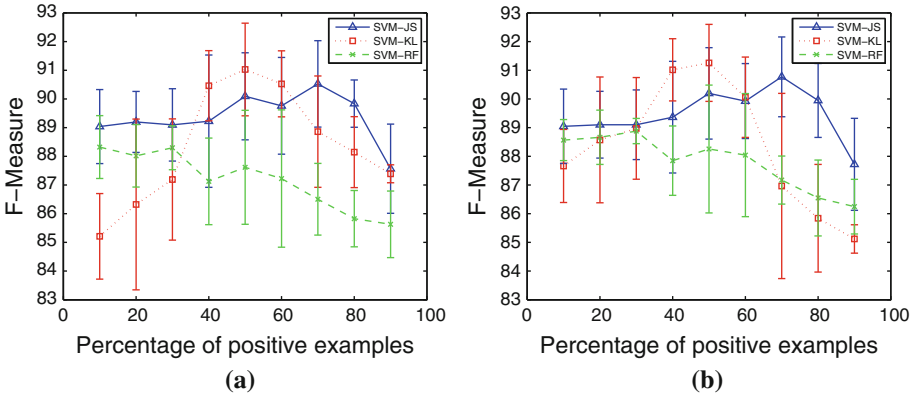

Fig. 17 F-measure for positive and negative classes on the kitchen domain. **a** F-measure+, **b** F-measure−

4. rf consistently performed the worst for positive:negative ratios greater than 3:7. For ratios less than 3:7, rf beat KL most of the time, but lost to JS in 3 of the 4 domains. rf, on the other hand, tends to performed worse as the positive class start to dominate. This is expected because rf only boosts positive class weights. As the negative class size shrinks, terms that appear predominantly in the positive class will increase, while those that appear predominantly in the negative class will shrink or weighted constantly otherwise. This has an overall effect of classifying everything as positive due to the abundance of positive class features.

For the negative class performances shown in Figs. 14b, 15b, 16b, and 17b, we can draw similar conclusions as follows:

1. JS consistently beat KL at extreme small proportion of negative class sizes, that is, at positive/negative ratios of 7:3, 8:2, 9:1 (toward the right).
2. KL performance peaked at the 5:5 ratio. KL was the best performer in 2 of the 4 domains for positive/negative ratios of 1:9 to 6:4.
3. rf deteriorates with decreasing negative class size. At extremely small positive class size (large negative class size, toward the left), rf performed comparably to the other two methods. This is because for terms occurring exclusively in the negative class, rf defaults to tf weights. Keep in mind that rf is biased toward weighting exclusively positive terms, but treat the exclusively negative terms by assigning constant tf weights. Thus, SVM was able to work on the tf weights from rf reasonably well as long as there are sufficient negative training samples (left side of axis). Moreover, as expected, rf negative class performance decreased as the positive class size rises. As described earlier, as positive class samples dominate, the classifier will classify everything as positive, leading to worsening negative class performance.

In summary, SVM–JS consistently outperforms SVM–KL for extremely small class size (highly imbalanced datasets). As the class distribution approaches uniform, SVM–KL can be marginally better than SVM–JS. These results are consistent to those reported in Sect. 4.3. Moreover, the performance of SVM–JS is more stable for all classes and across all class sizes compared to SVM–KL; SVM–JS performance is quite consistent with respect to varying amounts of examples from 10 to 90 %, whereas SVM–KL tends to perform well only for the negative class. In real-life applications, SVM–JS should thus be the all round better choice.

5 Conclusion

In this paper, we study the theoretical properties of the class feature centroid (CFC) classifier by considering the rate of change of each prototype vector with respect to individual dimensions (terms). Borrowing the e-folding time scale function from physics and biology, we show that classical centroid-based classifiers like arithmetic average centroid (AAC) and CGC have an inverse rate of change, while CFC has an exponential rate of change, proportional to the class document frequency. The implication of our theoretical finding is that CFC is inherently biased toward large (dominant majority) classes, which means it is destined to perform poorly for highly imbalanced data. Another practical concern about CFC lies in its overly aggressive design in weeding out terms that appear in all classes.

To overcome these CFC limitations while retaining its intrinsic and worthy design goals, we proposed an improved centroid-based classifier that uses precise term-class distribution properties instead of simple presence or absence of terms in classes. Specifically, terms are weighted based on the Kullback–Leibler divergence measure between pairs of class-conditional term probabilities; we call this the CFC–KL centroid classifier. We then generalized CFC–KL to handle multi-class data by replacing the KL factor with the multi-class Jensen–Shannon factor, with each class proportionally weighted by a soft-max function. We call this generalized approach CFC–JS.

Our proposed approach has been evaluated on 5 datasets. Experimental results show that the CFC–KL method consistently outperforms other methods on binary datasets. For highly imbalanced binary datasets, CFC–JS further outperforms CFC–KL and is more robust over various degrees of class-imbalance. To test the individual effectiveness of JS and KL term weighting approaches, we further evaluate the performance of SVM classifiers on highly imbalanced binary classification problems. We find that both SVM–JS achieves consistently better performance for all classes, while SVM–KL achieve significantly better performance only for the majority class. We thus conclude the SVM–JS is the better all-rounded choice for highly imbalanced datasets.

In general, we find that CFC when combined with KL (two class) or JS (multi-class) weighting will significantly boost classification performance, bringing centroid classifiers up to the accuracy level of SVM classifiers. For applications where training time is critical, such as large-scale online classification tasks, CFC combined with KL or JS term weighting becomes a viable option. Apart from simplicity and efficiency, CFC-based classifiers enjoy the advantage of being incrementally updateable, without the need for expensive re-training like in SVM. For future work, we will investigate the performance of CFC–JS and SVM–JS on true multi-class datasets instead of using one-versus-all settings.

Acknowledgments This research was supported in part by Singapore Ministry of Education's Academic Research Fund Tier 2 grant ARC 9/12 (MOE2011-T2-2-056).

References

1. Ali SM, Silvey SD (1966) A general class of coefficients of divergence of one distribution from another. J R Stat Soc 28:131–142
2. Bekkerman R, Gavish M (2011) High-precision phrase-based document classification on a modern scale. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '11, ACM, New York, pp 231–239
3. Blitzer J, Dredze M, Pereira F (2007) Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: The association for computer linguistics (ACL)

4. Bruce RF, Wiebe JM (1999) Recognizing subjectivity: a case study in manual tagging. Nat Lang Eng 5:187–205
5. Cover T, Hart P (2002) Nearest neighbor pattern classification. Knowl Based Syst 13:373–389
6. Debole F, Sebastiani F (2003) Supervised term weighting for automated text categorization. In: Proceedings of the 2003 ACM symposium on applied computing. SAC '03, ACM, New York, pp 784–788
7. Géry M, Largeron C (2011) BM25t: a BM25 extension for focused information retrieval. Knowl Inf Syst 32:1–25
8. Guan H, Zhou J, Guo M (2009) A class-feature-centroid classifier for text categorization. In: 18th international world wide web conference, pp 201–201
9. Han E-H, Karypis G (2000) Centroid-based document classification: analysis and experimental results. In: PKDD '00: Proceedings of the 4th European conference on principles of data mining and knowledge discovery. Springer, London, pp 424–431
10. Hatzivassiloglou V, McKeown KR (1997) Predicting the semantic orientation of adjectives. In: Proceedings of the eighth conference on European chapter of the association for computational linguistics. Association for Computational Linguistics, Morristown, pp 174–181
11. Hu M, Liu B (2004) Mining and summarizing customer reviews. In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, pp 168–177
12. Joachims T (2001) A statistical learning model of text classification with support vector machines. In: Proceedings of ACM SIGIR, pp 128–136
13. Junejo KN, Karim A (2012) Robust personalizable spam filtering via local and global discrimination modeling. Knowl Inf Syst 1–36. doi:10.1007/s10115-012-0477-x
14. Lan M, Tan CL, Su J, Lu Y (2009) Supervised and traditional term weighting methods for automatic text categorization. Pattern Anal Mach Intell 31:721–735
15. Langley P, Iba W, Thompson K (1992) An analysis of bayesian classifiers. In: AAAI '92: Proceedings of the tenth national conference on artificial intelligence. AAAI Press, pp 223–228
16. Lavesson N, Boldt M, Davidsson P, Jacobsson A (2011) Learning to detect spyware using end user license agreements. Knowl Inf Syst 26(2):285–307
17. Lewis DD (1998) Naive (bayes) at forty: The independence assumption in information retrieval. In: ECML '98: Proceedings of the 10th European conference on machine learning. Springer, London, pp 4–15
18. Lin J (1991) Divergence measures based on the shannon entropy. IEEE Trans Inf Theory 37:145–151
19. Liu W, Wang T (2011) Online active multi-field learning for efficient email spam filtering. Knowl Inf Syst 1–20. doi:10.1007/s10115-011-0461-x
20. Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press, Cambridge
21. Martineau J, Finin T, Joshi A, Patel S (2009) Improving binary classification on text problems using differential word features. In: Proceeding of the 18th ACM conference on information and knowledge management. CIKM '09, ACM, New York, pp 2019–2024
22. McCullagh P, Nelder JA (2000) Generalized linear models. Champman and Hall/CRC, New York
23. Nguyen TT, Chang K, Hui SC (2011) Supervised term weighting for sentiment analysis. In: Intelligence and security informatics
24. Nguyen TT, Chang K, Hui SC (2011) Word cloud model for text categorization. In: Proceedings of the 11th IEEE international conference on data mining, pp 487–496
25. Pang B, Lee L (2004) A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: Proceedings of the ACL
26. Pang B, Lee L (2005) Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the ACL
27. Quinlan JR, Rivest RL (1989) Inferring decision trees using the minimum description length principle. Inf Comput 80(3):227–248
28. Sebastiani F (2002) Machine learning in automated text categorization. ACM Comput Surv 34(1):1–47
29. Wang B, Japkowicz N (2010) Boosting support vector machines for imbalanced data sets. Knowl Inf Syst 25:1–20. doi:10.1007/s10115-009-0198-y
30. Wei F, Liu S, Song Y, Pan S, Zhou MX, Qian W, Shi L, Tan L, Zhang Q (2010) Tiara: a visual exploratory text analytic system. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining KDD '10, ACM, New York, NY, USA, pp 153–162

Author Biographies

Tam T. Nguyen received a B.E. degree and a M.E. degree from University of Technology, Hochiminh City, Vietnam, in 2001 and 2007, respectively. From 2001 to 2006, he had been working for Intergraph South East Asia. He is currently a Ph.D. student at the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include data mining, machine learning, and information retrieval.

Kuiyu Chang is an Assistant Professor of Computer Engineering at Nanyang Technological University, Singapore. Prior to that, he served as senior risk management analyst for ClearCommerce (now eFunds). From 2000 to 2002, Kuiyu was a member of technical staff at Interwoven (now Autonomy). He has served as program co-chair for PAISI (2006–2008), and publications chair for PAKDD 2006. Kuiyu has published over 50 papers, including in top journals (IEEE PAMI) and conferences (SIGIR, IJCAI, ICDE, ICDM, SDM), and is also recipient of 2 international best paper awards. Since 2005, he has been leading the ROSE (Review and Opinion Search Engine) project, which aggregates online Chinese sentiments. Kuiyu consults regularly for IT companies in China, Singapore, and Malaysia. He received his Ph.D. from the University of Texas at Austin, M.S. from the University of Hawaii at Manoa, and B.S. from National Taiwan University, all in Electrical/Computer Engineering.

Siu Cheung Hui is currently an Associate Professor in the Division of Information Systems, School of Computer Engineering, Nanyang Technological University, Singapore. He received his B.Sc. degree in Mathematics in 1983 and D. Phil in 1987 from the University of Sussex, UK. He worked in IBM China/Hong Kong Corporation as a system engineer from 1987 to 1990. His current research interests include data mining, Web mining, Semantic Web, intelligent systems, information retrieval, intelligent tutoring systems, timetabling, and scheduling.